Naval Research Laboratory

Stennis Space Center, MS 39529-5004

NRL/MR/7320--07-9082

Makef22: An ADCIRC Model Fort.22 Input File Creation Tool for Surface Wind and Pressure Forcing

CHERYL ANN BLAIN

Ocean Dynamics and Prediction Branch
Oceanography Division

ROBERT S. LINZELL

Planning Systems Incorporated
Stennis Space Center, Mississippi

BRETT ESTRADE

Center for Computation and Technology
Louisiana State University
Baton Rouge, Louisiana

December 7, 2007

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
07-12-2007	Memorandum Report	

4. TITLE AND SUBTITLE

Makef22: An ADCIRC Model Fort.22 Input File Creation Tool for Surface Wind and Pressure Forcing

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)

Cheryl Ann Blain, Robert S. Linzell,* and Brett Estrade†

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
73-6801-07-5

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Oceanography Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7320--07-9082

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
One Liberty Center
875 North Randolph Street
Arlington, VA 22203-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*Planning Systems Incorporated, MSAAP Building 9121, Stennis Space Center, MS 39529
†Center for Computation and Technology, Louisiana State University, Baton Rouge, LA 70808

14. ABSTRACT

The User's Manual for the Makef22 software tool, makef22.pl, contains descriptions of the software, including its functionality and usage. The Makef22 software is a Perl-based program that creates an ADvanced CIRCulation (ADCIRC) Model Surface Wind and Pressure Forcing file (fort.22) using the NWS = 2 option. The software was developed for ADCIRC version 45.11. In addition to the creation of fort.22 files, the Makef22 utility also can read and process an existing fort.22 file in order to 1) elongate the record either by adding zero valued records at the beginning of the fort.22 file or by repeating the first time record a specified number of times, and/or 2) ramp a specified portion of the fort.22 records from zero to full-scale values. This utility was designed so that readers for new data sources can easily be developed and incorporated into the Makef22 utility. Surface wind and pressure data sources on rectangular, regular girds at known times are read and interpolated onto an ADCIRC finite element mesh (FEM) at user-specified times. The results can be stored in an ASCII text, ADCIRC model specific, fort.22 forcing file.

15. SUBJECT TERMS

ADCIRC model	Makef22
Wind and pressure forcing	fort.22

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Cheryl Ann Blain
Unclassified	Unclassified	Unclassified	UL	41	**19b. TELEPHONE NUMBER** *(include area code)* (228) 688-5450

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

1. Description

a. Overview

The *makef22* utility is a Perl-based program that creates a surface wind and pressure forcing file in a format appropriate for the ADvanced CIRCulation (ADCIRC) model using the NWS = *2* option, e.g. the *fort.22* file (see Appendix I for details on the *fort.22* file description and format or the ADCIRC model online manual at http://www.adcirc.org). The utility also can read and process an existing *fort.22* file in order to 1) elongate the record either by adding zero valued records at the beginning of the *fort.22* file or by repeating the first time record a specified number of times, and/or 2) ramp a specified portion of the *fort.22* records from zero to full-scale values. This utility was designed so that readers for new data sources can easily be developed and incorporated into the *makef22* utility. Surface wind and pressure data sources on rectangular, regular grids at known times are read and interpolated onto an ADCIRC finite element mesh (FEM) at user-specified times. The results can be stored in an ASCII text, ADCIRC model specific, *fort.22* forcing file.

The *makef22.pl* utility is comprised of five components: the driver (*makef22.pl*), the date iterator (*Iterator.pm*), the input parameter file reader (*GetUserInput.pm*), the reader (e.g., *navo_gmt.pm*), and the processing program (*read_expand_ramp_f22_v3.F*). The user executes the driver, the input parameter file reader loads the user-supplied parameters from an ASCII text file (see Appendix II for an example of this file), the date iterator is used to facilitate iteration over a specified range of dates, and the reader facilitates the reading and interpolation of the surface wind and pressure data. The processing program is a standalone program that is invoked by the driver if the user chooses the processing option.

b. Methodology

The reader, such as *navo_gmt.pm*, which is used for reading the Generic Mapping Tools (GMT) network Common Data Form (netCDF) file format used by NAVOCEANO, is a Perl script called by the driver, *makef22.pl*. The reader serves as a wrapper around a data reader written in Fortran 90. While the actual reading of the data is handled by the Fortran 90 code, the wrapper handles the rest of the processing details such as compiling the Fortran 90 code into an executable (done once at the start if no binary executable program is supplied - *unsupported at this time*), locating the requested data files, and calling the reader executable for each date/hour that is requested. The Perl wrapper also tracks the number of files read, and manages the proper formatting of the data for the *fort.22* file based on what record is being written. The reader assumes one time record per file.

The Fortran 90 program contained in the reader, e.g. *navo_gmt.pm*, to be supplied as a standalone binary executable program, reads in surface wind velocity or wind stress components and/or surface pressure for any region supported by the reader (see Appendix V for a list of currently supported atmospheric models and regions). The fields are then passed back to the *makef22.pl* driver. Wind stress, if computed, is accomplished using the formula of Garratt (1977). Wind stresses are in units of *Pressure/Length*^{2} or in SI,

Manuscript approved October 11, 2007.

N/m^2, and are divided by the reference density of water to get units of $Length^2/Time^2$ (SI or metric units are determined by the units of $\boldsymbol{g}$ specified in the ADCIRC fort.15 parameter file). The surface pressure is represented by units of $Pressure/Length^2$ or in SI, N/m^2.

The data are then interpolated onto the ADCIRC mesh contained in a user-supplied *fort.14* file. Typically wind and pressure data sources are defined over rectangular regions using regularly-spaced points. The ADCIRC mesh is typically composed of irregularly shaped triangles whose vertices or nodes are also irregularly spaced. Interpolation of the wind and pressure data from a rectangular grid onto the FEM is performed using simple bilinear interpolation. Because of the need to interpolate the data onto the FEM, an existing ADCIRC *fort.14* file for the region of interest is required.

It is important to note that the Fortran 90 code that forms the data reader contained in *navo_gmt.pm* is a standalone program that can be copied into its own Fortran 90 source file and compiled using any Fortran 90 compiler. The resulting executable can be used by itself, but the user must provide the required input.

A recently added option allows the user to specify an existing, standalone, binary executable program that the reader invokes in place of the reader-contained Fortran 90 source code. This provides a means by which the user can avoid having to modify the Perl source code to add compiler parameters, or determine the correct compiler parameters when invoking the *makef22.pl* utility. This option also avoids program compilation every time the utility is executed, which occurs if no binary executable is supplied. This option is presently the only supported and recommended mode of operation for the data reader software.

Another recently added option allows the user to employ the binary executable processing program, *read_expand_ramp_f22_v3*, to process an existing *fort.22* file. There may be instances where the user wishes to have the non-zero wind and pressure forcing start at a time later than at the start of an ADCIRC model simulation. For example, a long ramp-up period may be necessary for the application of tidal forcing but wind and pressure forcing may not available during the entire ramping phase. In such cases, the wind and atmospheric pressure fields could be initially set to zero or some background value, or the first available wind and pressure record could be repeated for a specified length of time during model spin-up period. This type of accommodation is often necessary when applying wind and pressure forcing within the ADCIRC model (NWS not equal to 0), since the ADCIRC model expects wind and pressure forcing to extend from the start to the end of a simulation. In the above scenario, where non-zero wind and pressure forcing fields are applied after the ramp duration (DRAMP) that occurs internal to the ADCIRC code, the initial meteorological forcing values would also need to be gradually increased to full scale (i.e., “ramped”) to avoid the generation of numerical artifacts but must be ramped external to the ADCIRC code; this external ramping is handled by the supplied processing program. Figure 1 is a conceptual sketch illustrating the application of features available in the processing program, *read_expand_ramp_f22_v3*.

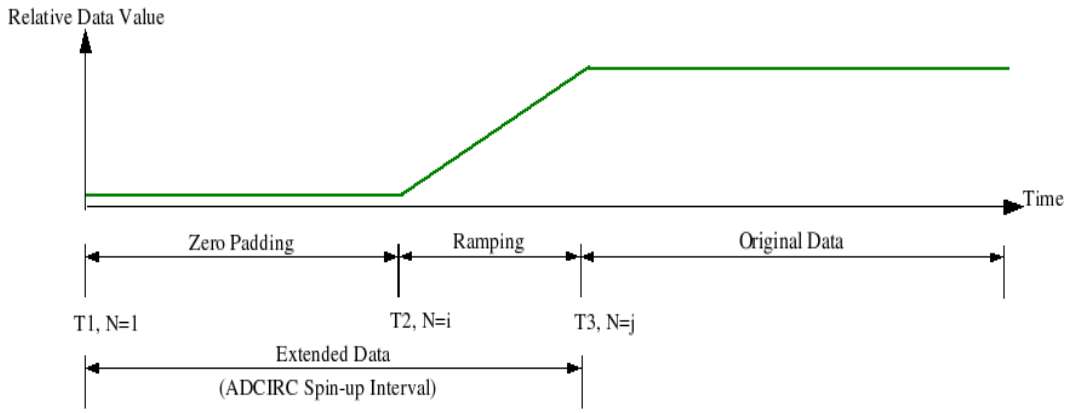

Figure 1. Sketch depicting the extension of meteorological data through the use of zero-padding, repetition of the first data record and then ramping of the extended data records, all accomplished via the read_expand_ramp_f22_v3 utility.

In the above figure, the green line represents an idealized time series of a meteorological forcing data value, such as atmospheric pressure. At time T1 (time step N=1), which would correspond to the start time of the model run, the data values are set to zero, zero-padding (or a background value, such as 10 m for atmospheric pressure). This initial value is maintained up to time T2 (time step N=*i*). From time T2 to T3 (N=*j*), the first non-zero data value is copied, and a ramp function is applied. Over this time interval, the data gradually increases to full value at the end of the ramp period, T3.

In this example, the original *fort.22* data values extend from the time step following T3 (N=*j+1*) to the end of the time series. Thus the original data set has been extended by the sum of the zero-padded interval (T1 to T2, *i* time steps) and the ramped interval (T2 to T3, *j-i* time steps). During an ADCIRC model run, the ramp period for the forcing of interest would correspond to the time interval T1 to T3 (*j* time steps). The time steps in this example refer to the data interval of the original meteorological data, e.g., three hours. The *fort.22* files contain no time information but require a constant data interval for the entire record. Hence, input to the processing program, is provided as numbers of time steps instead of hours or some other time increment.

c. Components

The "*makef22*" utility that creates the *fort.22* file is actually composed of eight primary modular parts as depicted in Figure 2:

1. *Iterator.pm* - a date-iterating Perl module that cycles over a user-specified date range
2. *GetUserInput.pm* – a Perl module that reads the user-supplied input parameter file, *makef22.pl.in*
3. *makef22.pl* - the main driver that the user executes
4. *navo_gmt.pm* - a *makef22.pl*-compatible data reader specifically designed to read

NAVO GMT netCDF files. All readers are specified by the "`reader`" option of *makef22.pl.in*

5. *nrl_coamps.pm* – a *makef22.pl*-compatible data reader specifically designed to read NRL binary COAMPS files.
6. *nrl_nogaps.pm* – a *makef22.pl*-compatible data reader specifically designed to read NRL binary NOGAPS files.
7. *nrl_bin2xyz* – a standalone data reformatting tool that converts NRL binary COAMPS or NOGAPS output files to ASCII text format.
8. *read_expand_ramp_f22_v3* – the data processing program for extending and ramping, an existing *fort.22* data file.

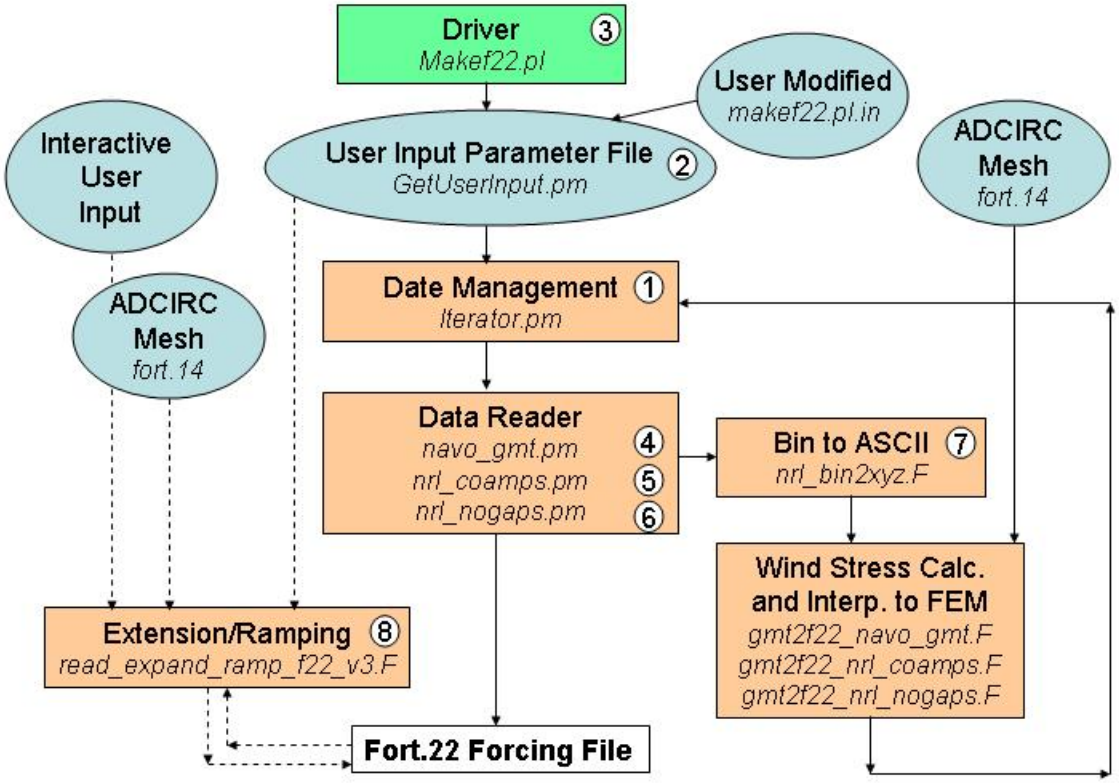

Figure 2. A schematic flow chart depicting the data flow and relationship between the software components comprising the Makef22 tool.

If the creation of a new *fort.22* file from user-supplied meteorological data is desired, the option to process an existing fort.22 file is disabled (see Table 1). Using the date iterator (*Iterator.pm*), the main driver (*makef22.pl*) iterates over a date range specified by user-defined starting and ending dates or a user-defined date range and hourly time increment. The date format is YYYYMMDDHH where YYYY indicates the year, MM the two-digit month, DD the two-digit day, and HH the two-digit hour using a 00-24 hour range. These input parameters are supplied in the ASCII text file, *makef22.pl.in* (see Table 1 and Appendix II), which are read at the program startup by the input file reader, *GetUserInput.pm*. Implementation of the date iterator is hidden from the user except for the initial specification of a date range and increment as indicated above. For each

date/hour in the range, the reader is called via a standard interface (see Appendix IV). The reader is responsible for returning the data for the requested date/hour in the proper *fort.22* format (see Appendix I).

If the processing mode is specified, data processing is enabled (see Table 1) and an existing *fort.22* file required (see §6). The main driver acquires the user inputs from the *makef22.pl.in* input parameter file via the input file reader. The data processing program (*read_expand_ramp_f22_v3*) is invoked to perform the extension and/or ramping of the original data found in the supplied *fort.22* file. A new file, named *fort.221*, is created during processing and is the final result of the *fort.22* processing. This file should be renamed to *fort.22*, after moving or renaming the original *fort.22* file, prior to execution by the ADCIRC model code.

The user can create more than one processed output file corresponding to different combinations of zero-padded or record repetition extension and ramping, bearing in mind that the output file (*fort.221*) should be moved or renamed after each execution of the *makef22.pl* driver. If several processing runs are planned, separate copies of the input parameter file (*makef22.pl.in)* should be created containing the various processing parameters. Then, for each execution of the driver, the parameter file of interest could be symbolically linked to *makef22.pl.in* to speed up or automate processing with a shell script. An example below demonstrates this approach:

```
# ls *.in   [A listing of three input files to be used for 3 different
             executions of makef22.]

      test1.in test2.in test3.in

# ln -fs test1.in makef22.pl.in     [Create symbolic link from test1.in
                                     to makef22.pl.in.]

# makef22.pl      [First execution of makef22.pl using test1.in input.]

# ln -fs test2.in makef22.pl.in     [Create symbolic link from test2.in
                                     to makef22.pl.in.]

# makef22.pl      [Second execution of makef22.pl using test2.in input.]

# ln -fs test3.in makef22.pl.in     [Create symbolic link from test3.in
                                     to makef22.pl.in.]

# makef22.pl      [Third execution of makef22.pl using test3.in input.]
```

d. Software Prerequisites

Perl must be installed on the platform on which the *Makef22* software will be installed. On Windows-based systems, the latest version of Perl from ActiveState Software Inc. (www.activestate.com) is recommended. The Generic Mapping Tools (GMT) freeware package (gmt.soest.hawaii.edu) and the network Common Data Form (netCDF) software (www.unidata.ucar.edu/packages/netcdf) must be installed for the Fortran programs to correctly operate. An existing ADCIRC Grid and Boundary Information File (fort.14) must also be present in the current working directory.

An external, standalone, pre-compiled Fortran data reader, such as *gmt2f22_navo_gmt*, must be present and applicable to data reformatting and interpolation of the desired meteorological data source. Additionally, the NRL Navy Operational Global Atmospheric Prediction System (NOGAPS) and Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS®) pre-compiled binary file reformatting program, *nrl_bin2xyz*, must be present to process those data types. Although the software is platform-independent, the standalone Fortran programs must be compiled for each platform on which they are installed.

e. Software Installation

The *makef22.pl* utility is located in the ADCIRC_Utilities repository in the Makef22 directory. The directory has the following contents:

- CONTENTS
- (directory) Documentation/
    - CONTENTS
    - ADCIRC_MANUAL_MakeF22_FINAL_09-25-07.doc
    - *README*
    - *README_Ramping*
- *GetUserInput.pm*
- *Iterator.pm*
- *Makefile*
- *Makefile.MSRC.romulus*
- (directory) *Sample_Input_Files/*
    - CONTENTS
    - *MASTER_makef22.pl.in*
    - *makef22.pl.in*
    - *makef22_generate.pl.in*
    - *makef22_ramping.pl.in*
- *fort22_reformat.awk*
- *makef22.pl*
- *read_expand_ramp_f22_v3.F*
- (directory) *readers/*
    - CONTENTS
    - *Makefile*
    - *Makefile.nrl_bin2xyz*
    - *cmplrflags.mk*
    - *config.guess*
    - *gmt2f22_navo_gmt.F*
    - *gmt2f22_nrl_coamps.F*
    - *gmt2f22_nrl_nogaps.F*
    - *grid2gmt.csh*
    - *hms2itm.f, idateadd.f, idayyr.f, idcen2idt.f, idt2idcen.f, idt2ymd.f, itm2hms.f, and ymd2idt.f* (external source code files for nrl_bin2xyz.f)
    - *navo_gmt.pm*
    - *nrl_coamps.pm*

- *nrl_nogaps.pm*
- *gmt2f22_navo_gmt.F*
- *gmt2f22_nrl_coamps.F*
- *gmt2f22_nrl_nogaps.F*
- *nrl_bin2xyz.f*
- *run_cmd.pl*

Step 1: Pre-Compilation of Fortran Programs

a) The Data Reader

The currently supported option is to pre-compile the supplied Fortran reader code (*gmt2f22_navo_gmt.F*) on the platform on which the software has been installed, a "makefile" has been supplied. The makefile, named *Makefile*, uses the GNU *make* program to compile the software on a variety of platforms. The accompanying shell script, *config.guess*, and the ASCII text compiler flag file, *cmplrflags.mk*, also are used by *make* to properly compile the software on any of several tested platforms (e.g., Linux 32-bit, Linux 64-bit, or IBM AIX). The *cmplrflags.mk* file also contains paths to the NetCDF library and include directories. The user should inspect this file prior to compilation to verify whether the compiler flags and netCDF paths are correct. The file can be edited with a text editor should any of the parameters require modification. To build the standalone binary executable program for the data reader, the following command is used:

```
# make gmt2f22_navo_gmt
```

On the IBM under AIX, the *gmake* program may be required, instead of *make*. This procedure should be performed only once after the software has been installed. The binary executable program, *gmt2f22_navo_gmt*, will then be used when the *makef22.pl* utility is invoked, so that the Fortran source code will not be compiled each time.

b) The NRL Data Reformatting Tool

In a similar manner, the standalone NRL binary data reformatting tool, *nrl_bin2xyz.F*, should be compiled using the supplied makefile using the following syntax:

```
# make -f Makefile.nrl_bin2xyz
```

Note that this tool is required for processing NRL meteorological binary data files, but has been tested only under Linux as of this writing. Users needing to use data types other than the NRL binary or NAVOCEANO GMT netCDF formats can contact the authors for further information.

c) The *Fort.22* Processing Program

The *fort.22* processing program, *read_expand_ramp_f22_v3.F* should be compiled using the supplied *Makefile* with the system *make* utility. An alternative makefile, *Makefile.MSRC.romulus*, also is supplied for use on IBM AIX-based systems.

Step 2: Configuration of NRL Binary Data Readers

If you wish to use the supplied readers for NRL binary COAMPS or NOGAPS data (*nrl_coamps.pm* or *nrl_nogaps.pm*, respectively), the readers must be modified to use the correct path to the GMT binary executable program, *xyz2grd*. These two programs are written in Perl, so they can be modified using any text editor. The lines containing the following code must be modified:

```
$self->{_XYZ2GRD} = '/common/utilities/GMT3.4.2/bin/xyz2grd';
```

so that "/common/utilities/GMT3.4.2/bin" is changed to the correct path for the system on which the software is installed. In *nrl_coamps.pm*, this code is at line 161, and in *nrl_nogaps.pm*, line 158.

(OPTIONAL) Step 3: An Environmental Variable for Alternative Data Readers

You may set an environmental variable, **`F22_READER_DIR`**, that specifies a reader directory other than the default location in the ADCIRC_Utilities/Makef22/readers directory. This directory is where all readers could be organized if multiple custom readers are available. Alternatively, one could specify a path for the variable, "readerdir", on line 5 of the *makef22.pl.in* file (see Appendix II). Examples for setting the environmental variable, **`F22_READER_DIR`**, in c-shell (csh) and Bourne shell (sh) are as follows:

csh: `setenv F22_READER_DIR "/path/to/readerdir/"`
sh: `set F22_READER_DIR="/path/to/readerdir";export F22_READER_DIR`

(OPTIONAL) Step 4: An Environmental Variable for the Date Management Program

If you wish to place *Iterator.pm* in a separate directory, move it there, and specify the new location by including it in the environmental variable called "**`PERL5LIB`**". For more information regarding "**`PERL5LIB`**", please consult "`perlfaq8`" which should be accessible on any Unix-like platform with Perl 5 installed. This topic is addressed under the "`perlfaq8`" heading "How do I keep my own module/library directory?"

2. Fort.22 Creation: Using *makef22.pl*

a. Usage

To execute the *Makef22* utility, the Perl script *makef22.pl* is invoked from the command line as:

```
# makef22.pl > fort.22
```

where output from the *makef22.pl* code is directed to the file, *fort.22*, a meteorological forcing file for the ADCIRC code. In the above example, *makef22.pl* is invoked from the command line using the input parameters specified in the *makef22.pl.in* file. The result is the creation of an ASCII text *fort.22* file containing either NRL or NAVOCEANO meteorological data sources, which are stored in IEEE binary or GMT netCDF format,

respectively. Within the *makef22.pl.in* file (included in Appendix II), the start date/hour of the desired wind and pressure data (e.g., `2004011200`) is first specified followed by information about either the termination date of the data (e.g., `2004011400`) or the length of the data record to be read. The data to be read in this example will extend for 2 days (e.g., `2d`) at 6-hour increments (e.g., `6h`). The 6-hour increment corresponds to a wind time increment (WTIMINC) of 21600 seconds. The ADCIRC model will read new meteorological data every WTIMINC seconds. The file name for the ADCIRC grid to be used for interpolation is specified next (e.g., `fort.14`). Then the directory location for the data readers is given (e.g., `~/makef22pl/readers`), and the specific data reader is identified. For NAVOCEANO, the appropriate data reader is referred to as "navo_gmt" (e.g., `navo_gmt`). On the next line, one should provide the name of the pre-compiled binary file containing the data reader (e.g., `gmt2f22_navo_gmt`) . Next, the parameter specifications for the meteorological data are passed to the reader through the specified reader options, "`readeropts`". The reader options include data type, (e.g. "`-O PW`" to indicate pressure and wind velocities, the alias for the spatial domain that corresponds to a domain definition in the reader (e.g., `-domain CENT_AM_New`), and the directory location of the meteorological data (e.g., `-datadir /u/lev1/common/ADCIRC-TRAINING-I/NAVO_GMT_DATA`). If no precompiled binary reader program is specified, the computer architecture is detected automatically (*unsupported at this time*), and the appropriate compiler options are extracted from inside the reader source code file (*unsupported at this time*). However, if a custom setting is added to the compiler database, one can name the architecture using the use the optional "arch" line specification (e.g., `i686`) to override the architecture detected by the program. The remaining parameters specified in the *makef22.pl.in* pertain to the ramping and/or expansion of a fort.22 file. For details on the meaning and use of these parameters, see Table 1 or §6.

Note that the number of records required for a simulation is determined by the simulation length (RNDAY) and the wind time interval (WTIMINC), both of which are defined in the *fort.15* settings file. For wind and/or pressure-forced ADCIRC simulations, forcing data is required for the entire length of the simulation. To determine the number of records needed, convert `RNDAY` to seconds (since those are the units of WTIMINC), and use the following formula:

$$num\_records = \mathrm{RNDAY}_{sec} / \mathrm{WTIMINC}_{sec} + 1$$

If the number is not a whole number, round up to guarantee that the simulation runs to completion. When specifying the date range and increment length, care must be exercised to ensure that enough records are created to span the entire simulation period. A general rule is to convert WTIMINC to hours, then set "`end`" to "{RNDAY_{day}}d{$\mathrm{WTIMINC}_{hr}$}h" and "`inc`" to "{$\mathrm{WTIMINC}_{hr}$}h". Using this approach and rounding up to obtain the required *num_records* will ensure that a sufficient number of wind/pressure data records are processed to extend for the entire length of the run.

b. Specifications

The file actually executed by the user is the driver, *makef22.pl*. The input parameters for this program, contained in the input file *makef22.pl.in* (see Appendix II), are defined in Table 1. All details regarding the meteorological data, such as the location and naming convention of the data files, are handled by the reader software. By design, the reader is relied upon to return a properly formatted *fort.22* record (see Appendix I) for the specified date/hour. No validation is done by the driver to ensure a correct data read. The driver simply sends the data returned from the reader directly to standard output. As such, the desired output must be redirected to a file using the operators ">" or ">>" (e.g., "`> fort.22`" as shown in the *makef22.pl* usage example above, §3a). The data can also be redirected into another program using the Unix pipe operator, "|".

Table 1. Input Parameters for makef22.pl (makef22.pl.in)

Parameter	***Req***	***Description***	***Valid values***
`start`	`Y`	Start date	`YYYYMMDDHH (required format)`

`end`	`Y`	End date; an actual date can be specified or a relative period of time after the start date. The values described in the "Valid Values" column can be combined. For example, "2m3d4h" will end 2 months, 3 days, and 4 hours after the start date.	`YYYYMMDDHH, or`
			`#h – hours past start,`
			`#d – days after start,`
			`#m – months after start,`
			`#y – years after start`

`inc`	`Y`	Time increment; the values described in the "Valid Values" column can be combined.
		For example, "2m3d4h" will increment 2 months 3 days and 4 hours for each iteration.
			`#h – hours,`
			`#d – days,`
			`#m – months,`
			`#y – years,`

`reader`	`Y`	Data reader name.	Data reader name is the name of the reader file without the extension (.pl or .pm)
`readeropts`	`Y`	Data reader options; options must be enclosed in quotes.	Depends on reader. For *navo_gmt.pm* details, please see §4a; for *nrl_coamps.pm* and *nrl_nogaps.pm*, refer to §5a.
`readerdir`	`N`	Reader directory specification; Over-rides the "`F22_READER_DIR`" environmental variable. If the environmental variable is not set, and this option is not used, the default directory searched is "./readers".	Any valid Unix directory path
`readerbin`	`[N]`	OPTIONAL: Name of binary executable data reader program.	gmt2f22_navo_gmt (see §1e), or any valid Unix file name (program must already exist).

Parameter	***Req***	***Description***	***Valid values***
`arch`	`[N]`	OPTIONAL: Computer architecture specification and associated Fortran compiler information; passed to reader as an option flag of the same name. If set, the specified value will override the auto-detected one.	Architecture types as defined internally inside the reader. See §4c for specific details. *This option is presently unsupported.*
`f14`	`Y`	ADCIRC mesh onto which data is interpolated	Any valid Unix file path to a valid ADCIRC mesh file (*fort.14*)
`ramp`	`Y`	Processing Flag; indicates whether data generation or processing is to be done. Processing is described in §6.	Zero (0) if no processing will be done (i.e., a new *fort.22* file will be generated), or one (1) if processing will be done.
`norig`	`[N]`	OPTIONAL: The number of time steps from the original *fort.22* file to use.	Required only if ramp=1; must be 1 or greater. Enter a large number (e.g., 999) if the number is unknown.
`nback`	`[N]`	OPTIONAL: The number of background/zero-valued time steps that are pre-pended to the output file.	Required only if ramp=1; 0 for no zero-valued records, or greater than 0 to create zero-valued records.
`ncopy`	`[N]`	OPTIONAL: The number of copies of the first original data record (time step).	Required only if ramp=1; 0 or greater. If set to 0 and nramp > 0, ramping will be applied to zero-valued records &/or original data, depending on values of nstart & nramp.
`nramp`	`[N]`	OPTIONAL: The number of time steps over which to apply the ramping function.	Required only if ramp=1; set to 0 for no ramping (i.e., zero-valued records &/or extending only), or greater than 0 to apply ramping.
`nstart`	`[N]`	OPTIONAL: The time step at which to start applying the ramping function.	Required only if ramp=1; 1 or greater; ignored if nramp=0.

One example of the *makef22.pl.in* file is presented in Appendix II and found in the directory "Sample_Input_Files" (e.g., `./Sample_Input_Files`).

3. The Data Reader

a. Details

The data reader for the driver *makef22.pl* is designed so that the data source is accessed

and converted to the proper *fort.22* format without the driver's intervention. The driver merely makes a standard function call and in return receives some data that is assumed to be in the proper format.

The reader is passed a date string by the driver that specifies a particular date/hour and returns data for that date formatted appropriately for an ADCIRC *fort.22* file. The *makef22.pl* takes this data and sends it to standard output. Again, the output must then be directed into a file or piped to another application, otherwise, the resulting data will be written to the terminal screen instead of an output file.

For flexibility in accommodating a variety of as of yet unknown data formats, *makef22.pl* does nothing to verify that the correct data record is being returned or that it is in the correct format. The driver simply facilitates the calling of a generic reader over a specified range of dates. The reader alone is responsible for:

1. Reading the data in its native format for a specified date/hour; the date format passed to the reader can be changed by re-setting a line in *makef22.pl*, i.e.,
   `"$CMDLINEOPTS{FORMAT} = '%Y%m%d%k';"`
   Please see Appendix III for other date formatting options.
2. Reading the appropriate data (pressure and/or wind stress, or wind velocity) and processing if required (i.e., convert wind velocity components to wind stress)
3. Interpolating the data values to the specified ADCIRC grid
4. Formatting the data as required for an ADCIRC *fort.22* file (see Appendix I)
5. Returning the formatted data for the specified date/time to the driver via standard output

b. Creation of a New Reader

An application programming interface (API) was created to facilitate the creation of new data readers. There are three requirements that must be followed in order for a data reader to work:

1. The plug-in must be written or contained within a valid Perl file. Note that Fortran or C code can easily be incorporated into a Perl file to facilitate the actual reading of the data files as illustrated in the reader, *navo_gmt.pm*. Additionally, standalone, binary executable programs can be invoked in any of several ways by Perl.
2. The Perl file must implement the following functions (detailed descriptions are contained in Appendix IV):
3. *new(string readeropts)* – initializes the reader
4. *print_info(void)* – prints reader information
5. *get_record(datestring "YYYYMMDDHH")* – retrieves data record based on the date/hour timestamp
6. *finalize(void)* – performs any final tasks before finishing
7. The reader must return the final data to the driver via standard out

Since the *navo_gmt.pm* reader is already a working reader, it is recommended that the file *navo_gmt.pm* be used as a basis for creating new reader modules unless the data read procedures are dramatically different. The *navo_gmt.pm* reader demonstrates how a

reader module is to:

1. Handle initialization
2. Compile source code if needed
3. Look for data files
4. Interact with the source code executable if compiled
5. Return data to the driver

c. Explanation of the Fortran Source Code

The driver, *makef22.pl*, and the reader, e.g., *navo_gmt.pm*, are simply wrappers around a Fortran program that creates one *fort.22* record each time it is called. Because of this, the Fortran code, which contains all of the data reads and data processing required for the *fort.22* file, can be compiled and used external to *makef22.pl* for other applications if so desired. Should one wish to execute the Fortran program outside of the *makef22.pl* framework, the user must manually handle details that *makef22.pl* and the reader handle automatically. These details include:

1. If used, the data file for the desired date/hour containing the U-component of the wind velocity (or stress if used), must be copied to the current working directory (CWD) as a *fort.31* file.
2. If used, the data file for the desired date/hour containing the V-component of the wind velocity (or stress if used), must be copied to the CWD as a *fort.32* file.
3. If used, the data file for the desired date/hour containing the pressure data must be copied to the CWD as a *fort.33* file.
4. An ADCIRC grid file (*fort.14*) containing the FE mesh that the data is to be interpolated to must be copied into the CWD.

Additionally, when the Fortran program is executed, it requires nine values via standard input. The user will need to manually enter in the values at the start of the program, or enter them in an ASCII text file and use input redirection to provide the values to the program. An example of input redirection is as follows:

```
# gmt2f22_navo_gmt < infile
```

The input values are contained in the file named `infile`, each on a separate line in the order of input. These values, in order with the type of data expected, are as follows:

1. IMAX
   Number of time records per file; NAVO GMT netCDF files contain a single time record per file, so *navo_gmt.pm* assumes “1”
2. Use Stress Data Instead of Wind Velocity?
   “1” (yes), use pre-calculated wind stress data; “0” (no) will calculate wind stress from the wind velocity data
3. What record number?
   Enter an integer for which record is being written; the first two records of the *fort.22* file are written in a different format, and this value tells the reader which format to use.

4. DATE in *YYYYMMDD* Format
   Date for which this data file is valid
5. HOUR in *HH* Format
   Hour of DATE for which this data file is valid.
6. Use Pressure?
   "1" (yes), a pressure data file is read; "0" (no) pressure values are set to 0.0.
7. Pressure File Exists?
   "1" (yes), this flag tells the reader the pressure file exists; "0" (no), pressure is set to 0.0 at all points.
8. U-Component (wind or stress) File Exists?
   "1" (yes), this flag tells the reader the u-component wind file exists; "0" (no), the u-component wind file is set to 0.0 at all points.
9. V-Component (wind or stress) File Exists?
   "1" (yes), this flag tells the reader the v-component wind file exists; "0" (no), the v-component wind file is set to 0.0 at all points.

Because *makef22.pl* and the data reader pass values via standard output, the question prompts written to the screen are suppressed (i.e. commented out) in the Fortran program. It may be more helpful for the user to uncomment the question prompts requesting standard input when the program is run in standalone mode.

A *Makefile* and two accompanying files also are provided to enable the user to create a standalone, binary executable program. Details for creating a standalone, binary executable program are provided in §1e.

4. The NAVO GMT Wind Reader, *navo_gmt.pm*

a. Details

The *makef22* reader plug-in for the GMT netCDF data file format used by NAVOCEANO is called *navo_gmt.pm*, and is a Perl file that serves as a wrapper around a data reader written in Fortran 90. The Fortran 90 code for this reader is located at the end of the file, *navo_gmt.pm*, after the "**_DATA_**" section. While the actual reading of the data is handled by the Fortran 90 code, the wrapper handles the rest of the processing details such as compiling of the Fortran 90 code (done once at the start of each execution) if a binary executable is not specified, locating the requested data files, and calling the Fortran 90 data reader executable (or user-specified binary executable) to read each date/hour of data that is requested. The Perl wrapper also tracks the number of files returned, and manages the proper formatting of the data for the *fort.22*. Please note that since the NAVO GMT netCDF file contains a single time step of data per file, *navo_gmt.pm* tracks the number of files read. It assumes that there will be a single time step of data per file, and it uses this method to determine the formatting. (See Appendix I for the format of the *fort.22* file)

The Fortran 90 executable is designed to read specified data files for pressure or wind, compute wind stress if wind velocity fields are used, and interpolate the resulting values to the ADCIRC mesh contained in the *fort.14* file. The processed data is then printed to

standard output. The Perl wrapper captures this output, and redirects it to the driver, *makef22.pl*. In turn, the *makef22.pl* driver directs the data to its standard output where the user redirects the data to be written to an ADCIRC *fort.22* file. It is important to note that the Fortran 90 code contained in *navo_gmt.pm* serves as a standalone Fortran source file, compiled using any Fortran 90 compiler, and the resulting executable can be used alone to read and process the data. The wrapper merely takes care of the tedious task of running the Fortran 90 executable for each time increment needed to complete specified the time series of data.

The driver, *makef22.pl*, is designed to pass user-supplied options to the reader. These options are read from the *makef22.pl.in* file (see Table 1, §2b). Since these options can potentially change significantly from reader to reader, *makef22.p*l treats them as a string passed through the "`readeropts`" parameter. This string is then passed to the reader (e.g., *navo_gmt.pm*) when it is initialized.

The reader used for NAVO's GMT wind data requires certain flags as detailed in Table 2. Note that the reader is built for the *makef22* utility, thus it is not designed to be used outside of this context.

Table 2. Flag Specifications for the Data Reader, navo_gmt.pm

Flag	***Req***	***Description***	***Valid values***
`-O`	`[N]`	OPTIONAL: Data specification for creation of the *fort.22* file	`P` – pressure data only `PS` – pressure and pre-calculated wind stress (from a file) `W` – wind velocity data only (wind stress will be computed using Garratt, 1977) `S` – pre-calculated wind stress only (from a file) `PW` – (default) pressure and wind velocity data (wind stress will be computed using Garratt, 1977)
`-datadir`	`[N]`	OPTIONAL: Allows directory containing NAVO GMT data to be specified at run time.	Any valid Unix file path.
`-domain`	`Y`	Geographical domain name associated with data	Domain values are defined within the reader. See §4b for more details. A listing of currently supported domains can be seen in Appendix V.
`-arch`	`[N]`	OPTIONAL: Computer architecture specification and associated Fortran compiler information; passed onto reader as an option flag of the same name. If set, the specified value will override the auto-detected one. Passed in by *makef22.pl*	Architecture types as defined internally inside the reader. See §4c for specific details. *This option is currently unsupported.* Passed by *makef22.pl*

Flag	*Req*	*Description*	*Valid values*
`-f14`	Y	ADCIRC mesh onto which data is interpolated. Passed in by *makef22.pl*	Any valid Unix file path to a valid ADCIRC mesh file (*fort.14*). Passed by *makef22.pl*
`-help/-?`	[N]	OPTIONAL: Returns help information, then exits	N/A

b. Adding a New Domain to *navo_gmt.pm*

Domain information is stored locally in the reader, *navo_gmt.pm*, using a record structure that facilitates the addition of new records. There are several fields required to describe the data: an identifying reference name, the physical location, file naming convention, and dimensional parameters specific to the data domain. Many of these parameters are used to create the input file name based on the date, time, domain, and parameter of interest.

Information for each of the supported domains is stored in the subroutine "`get_domain_info`", found in *navo_gmt.pm*. The record structure has the following form in `sub get_domain_info`:

```
my %DOMAINS = (
     # Record 1
     DOMAIN1 => {
        ABBREV => "CENT_AM",
        COMMENT => "Gridded Central America kept at NAVO - production
settings",
        NUMTIMEREC_DEFAULT => '1',
        WLATMIN => '0.0',    # south
        WLATMAX => '32.0',   # north
        WLONMIN => '-120.0' ,# west
        WLONMAX => '-60.0',  # east
        DATA_ROOTS => "",
        SUFFIX  => "COAMPS_CENT_AM-fcst_ops-cent_am_nest2_appl-",
        PRESSURE => "pres-msl-0.000000-",
        U_WIND   => "wnd_ucmp-ht_sfc-10.000000-",
        U_STRESS => "wnd_strs_ucmp-surface-0.000000-",
        V_WIND   => "wnd_vcmp-ht_sfc-10.000000-",
        V_STRESS => "wnd_strs_vcmp-surface-0.000000-",
        TYPE => "RAW",
     }, # end Record 1
     # Record 2
     DOMAIN2 => {
       ...
     },# end Record 2
     # Record 3
     DOMAIN3 => {
       ...
     },
   );
```

Table 3 outlines the fields defined within the domain record listed above. The NAVO GMT data files are self describing by virtue of their netCDF format thus many of the fields specified in the domain record are unnecessary and not used. Also, some domains

present in the Perl source code may not have all of the fields shown above. The domain record attributes are shown as an example of how to include information for data sources that do not contain self-descriptive data like the GMT netCDF files. Please note that the Fortran 90 reader is specifically designed for the GMT netCDF files, and the use of differently formatted data will require modification to the Fortran 90 reader within *navo_gmt.pm* or the Fortran program, *gmt2f22_navo_gmt.F*.

The most important fields within the domain record for the *navo_gmt.pm* reader are ones that define the name and location of the data files. Note that all records defining a particular data domain must be contained within the "`%DOMAINS( ... )`" record structure.

Table 3. Description of the Domain Records Within navo_gmt.pm

Key	***Req***	***Description***	***Valid Value***
`ABBREV`	`Y`	Data domain name identical to that specified by the "`-domain`" flag in *navo_gmt.pm*	Any alphanumeric string without spaces
`COMMENT`	`[N]`	OPTIONAL: Description of the data, not used	Any string of characters
`NPTAU`	`[N]`	OPTIONAL: Number of time intervals found in pressure data files, not used	Integer
`NWTAU`	`[N]`	OPTIONAL: Number of time intervals found in wind velocity/stress data files, not used	Integer
`LAT`	`[N]`	OPTIONAL: E-W dimension of data grid points in latitude, not used	Integer
`LONG`	`[N]`	OPTIONAL: N-S dimension of data grid points in longitude, not used	Integer
`WLATMIN`	`[N]`	OPTIONAL: Minimum latitude of data domain, not used	Float (deg)
`WLATMAX`	`[N]`	OPTIONAL: Maximum latitude of data domain, not used	Float (deg)
`WLONMIN`	`[N]`	OPTIONAL: Minimum longitude of data domain, not used	Float (deg)
`WLONMAX`	`[N]`	OPTIONAL: Maximum longitude of data domain, not used	Float (deg)
`WLATINC`	`[N]`	OPTIONAL: Latitudinal resolution, not used	Float (deg)

Key	*Req*	*Description*	*Valid Value*
WLONINC	[N]	OPTIONAL: Longitudinal resolution, not used	Float (deg)
WTIMINC	[N]	Time record increment, not used	Integer (seconds)
DATA_ROOTS	Y	Search directories for data files; searched in order of entry; first match will be the data source used.	Valid file path (may be empty)
SUFFIX	Y	File naming convention; assumed common to all data file names in the domain	Alphanumeric string, no spaces
PRESSURE	Y	String used to identify pressure data files	Alphanumeric string, no spaces
U_WIND	Y	String used to identify U-component of wind velocity data	Alphanumeric string, no spaces
U_STRESS	Y	String used to identify U-component of wind stress data	Alphanumeric string, no spaces
V_WIND	Y	String used to identify V-component of wind velocity data	Alphanumeric string, no spaces
V_STRESS	Y	String used to identify V-component of wind stress data	Alphanumeric string, no spaces
TYPE	[N]	OPTIONAL: Identifies data format type, not used	Any alphanumeric string, no spaces

c. Adding New Compilers and Architectures to *navo_gmt.pm*

Some modifications to the data reader may be required to add compatibility for new computer architectures and their Fortran compilers. Additionally, the makefile (*Makefile*) and the compiler flag file, *cmplrflags.mk*, for the standalone binary executable program also may require modification to add compatibility with new architectures or compilers. The data reader obtains computer architecture and compiler information from a function called "`set_compiler_info`". The purpose of this function is to obtain information on the computer architecture and Fortran compiler associated with the computer platform specification invoked by the data reader using the reader's "`arch`" parameter. The information for each entry is stored in a data structure that is easily extended. Both the standalone reader and the embedded reader have been ported successfully to two computer platforms, the IBM SP4 (romulus), and an I686-Linux machine. These architectures use the Fortran 90 compilers "xlf90_r" and "pgf90," respectively, to compile the Fortran 90 code used to read the netCDF GMT files. The record structure in "`set_compiler_info`" has the following format:

```
@COMPILER_LIST = (
   # Record 1
   [{
     ARCH     => "",
     COMPILER => "",
     FLAGS    => "",
     ENVARS   => {ENVAR1=>'value',ENVAR2=>'value2'},
     FORMAT   => '%c %f %s -o %x',
    }
   ],
   # Record 2
   [{
     ARCH     => "",
     COMPILER => "",
     FLAGS    => "",
     ENVARS   => {ENVAR1=>'value',ENVAR2=>'value2'},
     FORMAT   => '%c %f %s -o %x',
    }
   ],
)
```

All records must be contained within the "`@COMPILER_LIST( ... )`" array whose keys are described in Table 4.

NOTE: Multiple architecture names may be specified under the "ARCH" key in a comma delimited list. This is to allow for the same settings to be reused, and is especially helpful for the automatic architecture detection since some platforms use the same settings, but the name derived from the detection routines may differ slightly.

Table 4. Description of the Compiler_List Flags in navo_gmt.pm

Key	***Req***	***Description***	***Valid Values***
`ARCH`	`Y`	Specifies the identifier used to name this architectural configuration. Passed to the reader using the "`-arch`" flag.	Alphanumeric string without spaces. Multiple names may be listed here, but must be separated by a comma. Example: "archtype1, archtype2, etc…"
`COMPILER`	`Y`	Specifies the compiler used to create an executable binary. Any Fortran compiler can be used as long as it has a command line interface.	The full path of the compiler being used to compile the Fortran code located at the end of *navo_gmt.pm* under "**`_DATA_`**".
`FLAGS`	`Y`	Specifies any compile time flags , libraries, linked files, etc.	Standard flags specifying library paths (-L), include directories (-I), libraries (-l), object files/source files to link to, etc. These flags are compiler specific.

Key	*Req*	*Description*	*Valid Values*
ENVARS	Y	For dynamic specification of environmental variables that are required at run time. For example, some compilers require an environmental flag to be set at compilation to indicate if data read/writes are big endian or little endian.	An anonymous hash must be specified here. Example: { ENVAR1=>'value', ENVAR2=>'value2' }, Creates environmental variables accessible to the compilation process named "ENVAR1" and "ENVAR2" with their respective values.
FORMAT	Y	Specify format for the compile command. This allows most any compiler with a command line interface to be used. The most common compilers can be specified using the format: "%c %f %s -o %x"	%c = compiler %s = source file %f = flags %x = executable

The best strategy to use when adding new compiler information is to determine how to compile the Fortran code as a standalone file, then use the insight gained on the specification of compiler options to fill in the record for the new compiler/architecture. If the standalone binary executable program is to be used, then modifications to the Perl source code described here would be optional. The modifications would then be applied to *cmplrflags.m*k and *Makefile*. For further assistance modifying these files, contact the authors.

5. The NRL COAMPS and NOGAPS Readers, *nrl_coamps.pm* and *nrl_nogaps.pm*

a. Details

The *makef22* reader plug-ins for the IEEE binary data file format used by NRL are called *nrl_coamps.pm*, for data from the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS®), and *nrl_nogaps.pm*, for data from the Navy Operational Global Atmospheric Prediction System (NOGAPS). These are Perl files that serve as wrappers around a Fortran 90 data reformatting tool, *nrl_bin2xyz*, the GMT ASCII text-to-netCDF reformatting tool, *xyz2grd*, and the previously described data reader written in Fortran 90 (see §4). The Fortran 90 code for this reader is located at the end of either Perl file, after the "**_DATA_**" section. While the actual reading of the data is handled by the Fortran 90 code, each wrapper handles the rest of the processing details such as compiling of the Fortran 90 code (done once at the start of each execution if a binary executable is not specified), locating the requested data files, and calling the Fortran 90 data re-formatter executable to reformat each data file that is requested. Each Perl wrapper also invokes

xyz2grd to reformat the data at each time step into netCDF format for the reader program, tracks the number of records returned, and manages the proper formatting of the data for the *fort.22*.

Note that since the NRL model output files contain multiple time steps of data per file, the readers track the date/time groups (DTGs) of the files that have been reformatted to ASCII text. The re-formatter creates a separate output file for each time step and parameter (i.e., atmospheric pressure and U- and V-components of wind speed or stress). The DTG corresponding to each model time step contains, respectively, the year, month, day, and hour. Each ASCII text output file name has the DTG embedded in it, so that the data for the requested date and time are correctly identified using the file name. The data for the current date and time are then converted into GMT netCDF format using the GMT reformatter (*xyz2grd*), and then the self-contained and automatically compiled reader or the user-supplied binary reader executable (*gmt2f22_navo_gmt* or equivalent) reads the netCDF files and returns the formatted output. The readers assume that there will be a single time step of data per converted file, and it uses this method to determine the formatting. (See Appendix I for the format of the *fort.22* file.)

If the user chooses a time increment that differs from the atmospheric data time increment, data at user time steps for which no atmospheric data exists are created using simple linear interpolation between the atmospheric data time steps which bracket the requested user time steps. If, for example, the user chooses to obtain forcing fields every hour but the model has output every three hours, then the readers interpolate to each hour that falls between the atmospheric data times.

The NRL COAMPS and NOGAPS atmospheric model repositories contain files archived from the 00Z and 12Z model runs. For each of these files, the analysis time (a "tau," or model time step, of 0 hr) corresponds to the model run time, and the forecasts (tau > 0) correspond to that number of hours past the model run time. If the user chooses a start time other than 00 or 12 hr, the readers determine the most recent model run time and start the data extraction from the corresponding output files. If, for example, the user chooses a starting DTG of 2006090103 (corresponding to September 1, 2006, at 03 hr), the most recent model run time would be 2006090100 (same date at 00 hr), and the model output files with that time stamp in the file names would be used to extract the data into separate ASCII text files for each time step. The user-supplied start time is used once the correct model output files are determined and the data are extracted and reformatted.

The Fortran 90 reader executable is designed to read specified data files for pressure or wind, compute wind stress if wind velocity fields are used, and interpolate the resulting values to the ADCIRC mesh contained in the *fort.14* file. The processed data is then printed to standard output. The Perl wrapper captures this output, and redirects it to the driver, *makef22.pl*. In turn, the *makef22.pl* driver directs the data to its standard output where the user redirects the data to be written to an ADCIRC *fort.22* file. It is important to note that the Fortran 90 code contained in *nrl_coamps.pm* or *nrl_nogaps.pm* serves as a standalone Fortran source file, compiled using a Fortran 90 compiler, and the resulting executable can be used alone to read and process the data. The wrapper merely takes care of the tedious task of running the Fortran 90 executable for each time increment

needed to complete specified the time series of data.

As of this writing, the Fortran code in these Perl modules is untested. The standalone, binary executable reader, *gmt2f22_navo_gmt*, has been used for initial development and testing.

The driver, *makef22.pl*, is designed to pass separate options to the reader. Since these options can potentially change significantly from reader to reader, *makef22.p*l treats them as a string passed through the "`readeropts`" parameter. This string is then passed to the reader (e.g., *nrl_coamps.pm*) when it is initialized.

The readers used for NRL model wind fields require certain options, contained in the *makef22.pl.in* file, as detailed in Table 5. Note that the readers are built for the *makef22* utility, and thus are not designed to be used outside of this context.

Table 5. Flag Specifications for the Data Readers, nrl_coamps.pm and nrl_nogaps.pm

Flag	***Req***	***Description***	***Valid values***
`-O`	[N]	OPTIONAL: Data specification for creation of the *fort.22* file	P – pressure data only PS – pressure and pre-calculated wind stress (from a file) W – wind velocity data only (wind stress will be computed using Garratt, 1977) S – pre-calculated wind stress only (from a file) PW – (default) pressure and wind velocity data (wind stress will be computed using Garratt, 1977)
`-datadir`	[N]	OPTIONAL: Allows directory containing NRL COAMPS or NOGAPS data to be specified at run time.	Any valid Unix file path.
`-domain`	Y	Geographical domain name associated with data	Domain values are defined within the reader. See §5b for more details. A listing of currently supported domains can be seen in Appendix V.
`-arch`	[N]	OPTIONAL: Computer architecture specification and associated Fortran compiler information; passed onto reader as an option flag of the same name. If set, the specified value will override the auto-detected one. Passed in by *makef22.pl*	Architecture types as defined internally inside the reader. See §5c for specific details. *This option is not currently supported.* Passed by *makef22.pl*
`-f14`	Y	ADCIRC mesh onto which data is interpolated. Passed in by *makef22.pl*	Any valid Unix file path to a valid ADCIRC mesh file (*fort.14*). Passed by *makef22.pl*

Flag	*Req*	*Description*	*Valid values*
-overwrite	Y	Indicates whether data at forecast times are to be overwritten by the next available model output file	0 – do not overwrite data; use the current model output file for the entire period 1 – overwrite data; use the next available model output file to provide data starting at the next analysis hour (00 or 12 hr)
-help/-?	[N]	OPTIONAL: Returns help information, then exits	N/A

b. Adding New Domains to *nrl_coamps.pm* and *nrl_nogaps.pm*

Domain information is stored locally in the readers using a record structure that is similar to that of *navo_gmt.pm*. There are several fields required to describe the data: an identifying reference name, the physical location, file naming convention, and dimensional parameters specific to the data domain. Many of these parameters are used to create the input file name based on the date, time, domain, and parameter of interest.

Information for each of the supported domains is stored in the subroutine, "get_domain_info." The record structure has the following form in sub get_domain_info:

```
my %DOMAINS = (
     # Record 1
     CEN_AMERICA => {
        ABBREV => "cen_america",
        SUFFIX  => "cen_amer",
        COMMENT => "",
        NPTAU => '3',         # Tau values (hr)
        NSTAU => '3',
        NWTAU => '3',
        NUM_PTAU => '17',     # Number of tau values
        NUM_STAU => '17',
        NUM_WTAU => '17',
        LAT     => '161',
        LONG    => '301',
        WLATMIN => '0.0',    #south
        WLATMAX => '32.0',   #north
        WLONMIN => '-120.0', #west
        WLONMAX => '-60.0',  #east
        WLATINC => '0.2',
        WLONINC => '0.2',
        WTIMINC => '10800',
        DATA_ROOTS => "/u/NOGAPS/COAMPSg/cen_amer",
        PRESSURE => "pres",
        U_WIND   => "wnd_ucmp",
        U_STRESS => "wnd_strs_ucmp",
        V_WIND   => "wnd_vcmp",
        V_STRESS => "wnd_strs_ucmp",
        # TYPE is defined above in the @ARCHIVE_TYPE array
        TYPE => "RAW",    }, # end Record 1
     # Record 2
     DOMAIN2 => {
       ...
     },# end Record 2
     # Record 3
```

```
    DOMAIN3 => {
      ...
    },
);
```

Table 6 outlines the fields defined within the domain record listed above. The NRL binary files have no geographical information stored internally. The binary file re-formatter and the GMT netCDF re-formatter each require several of the parameters, which the reader module provides during execution. Also, some domains present in the Perl source code may not have all of the fields shown above. The domain record attributes are shown as an example of how to include information for data sources that do not contain self-descriptive data like the netCDF files. Please note that the Fortran 90 reader is specifically designed for the netCDF files, and the use of differently formatted data will require modification to the Fortran 90 reader within *nrl_coamps.pm*, *nrl_nogaps.pm*, or to the source code in *gmt2f22_navo_gmt.F*.

The most important fields within the domain record for the readers are ones that define the name and location of the data files. Note that all records defining a particular data domain must be contained within the "`%DOMAINS( ... )`" record structure.

Table 6. Description of the Domain Records Within nrl_coamps.pm and nrl_nogaps.pm

Key	***Req***	***Description***	***Valid Value***
ABBREV	Y	Data domain name identical to that specified by the "`-domain`" flag in *nrl_coamps.pm* or *nrl_nogaps.pm* and required by *makef22.pl.in*	Any alphanumeric string without spaces
COMMENT	N	Description of the data, or list of model analysis and forecast times if non-uniform times are used	Any string of characters, or array of integers
NPTAU	Y	Time interval of pressure data files	Integer
NSTAU	Y	Time interval of wind stress data files	Integer
NWTAU	Y	Time interval of wind velocity data files	Integer
NUM_PTAU	Y	Number of time intervals found in pressure data files	Integer
NUM_STAU	Y	Number of time intervals found in wind stress data files	Integer
NUM_WTAU	Y	Number of time intervals found in wind velocity data files	Integer

Key	*Req*	*Description*	*Valid Value*
LAT	[N]	OPTIONAL: E-W dimension of data grid points in latitude, not used	Integer
LONG	[N]	OPTIONAL: N-S dimension of data grid points in longitude, not used	Integer
WLATMIN	[N]	OPTIONAL: Minimum latitude of data domain, not used	Float (deg)
WLATMAX	[N]	OPTIONAL: Maximum latitude of data domain, not used	Float (deg)
WLONMIN	[N]	OPTIONAL: Minimum longitude of data domain, not used	Float (deg)
WLONMAX	[N]	OPTIONAL: Maximum longitude of data domain, not used	Float (deg)
WLATINC	[N]	OPTIONAL: Latitudinal resolution, not used	Float (deg)
WLONINC	[N]	OPTIONAL: Longitudinal resolution, not used	Float (deg)
WTIMINC	[N]	OPTIONAL: Time record increment, not used	Integer (seconds)
DATA_ROOTS	Y	Search directories for data files; searched in order of entry; first match will be the data source used.	Valid file path (may be empty)
SUFFIX	Y	File naming convention; assumed common to all data file names in the domain	Alphanumeric string, no spaces
PRESSURE	Y	String used to identify pressure data files	Alphanumeric string, no spaces
U_WIND	Y	String used to identify U-component of wind velocity data	Alphanumeric string, no spaces
U_STRESS	Y	String used to identify U-component of wind stress data	Alphanumeric string, no spaces
V_WIND	Y	String used to identify V-component of wind velocity data	Alphanumeric string, no spaces
V_STRESS	Y	String used to identify V-component of wind stress data	Alphanumeric string, no spaces

Key	*Req*	*Description*	*Valid Value*
TYPE	[N]	OPTIONAL: Identifies data format type, not used	Any alphanumeric string, no spaces

c. Adding New Compilers and Architectures

Some modifications to the data readers may be required to add compatibility for new computer architectures and their Fortran compilers. Additionally, the makefile (*Makefile*) and the compiler flag file, *cmplrflags.mk*, for the standalone binary executable program also may require modification to add compatibility with new architectures or compilers. Refer to §5c for details. Note however this option is not currently supported. As previously mentioned, development and testing of these readers was performed using the binary executable reader option with *gmt2f22_navo_gmt*. This is the recommended option to use for normal *fort.22* data file generation.

6. Fort.22 Processing

a. Overview

Processing an existing *fort.22* data file is performed by the standalone, Fortran 90 program, *read_expand_ramp_f22_v3.F* or the binary executable program, *read_expand_ramp_f22_v3*. One option within the processing program is the expansion of the number of records prior to the beginning existing *fort.22* data file using a user-specified number of zero records (i.e., the wind velocity or stress U- and V-components are zero, and the atmospheric pressure is a "background" [i.e., ambient] value, such as 10 m^2/s^2). Another option is to expand the number of records at the beginning of the existing *fort.22* data file by making repeated copies of the initial time record. The processing program also controls the ramping of the data from zero or background values to full scale values over a user-specified time interval that is equivalent to some multiple of time steps. Typically ramping is applied over a series of repeated records based on copies of the initial data record in the original *fort.22* file. The processing program also has a feature by which one can extract a user-specified number of time steps from the original data, starting from the beginning of the data. Through extending and ramping, the original data can be expanded by the number of time steps corresponding to the specified spin-up period of an ADCIRC model run.

b. Details

A copy of the program, or a symbolic link to it, must reside in the user's CWD. The input parameters for the program, described in Table 1 in §2b, are provided in the input parameter file, *makef22.pl.in*. An example of this file is presented in Appendix II. The input data file (or symbolic link) must be named *fort.22*, and an ADCIRC grid file (or link), named *fort.14*, also must be present in the CWD. The output of the program is a *fort.22*-format file named *fort.221*. The user should know the number of time steps in the input *fort.22* file.

The program is interactive, but user input can be supplied using an input redirection file. An example of this method is as follows:

```
# read_expand_ramp_f22_v3 < ramp.in
```

where `ramp.in` is the name of the input file containing the required parameters. Alternatively, the Unix "here document" construct can be used, as shown in the following example:

```
# read_expand_ramp_f22_v3 << TheEnd
Param1
Param2
 ⋮
TheEnd
```

where the individual parameters (represented by `Param1`, `Param2`, and so on) are entered in the correct order, and the text string `TheEnd` is a flag indicating the end of input. The *makef22.pl* driver invokes the program using a Perl named pipe scheme, in the same manner as the data reader is invoked.

Each execution of the program produces a log file containing the local time and date of the run and a summary of the user inputs and resulting outputs for the *fort.221* file. The name of the log file is of the form *fort22ramp_YYYYMMDDHHMMSS.log* where *YYYYMMDDHHMMSS* is the current system date and time at the start of program execution. In particular, *YYYY* is the 4-digit year, *MM* is the 2-digit month number, *DD* is the day of the month, *HH* is the hour in 24-hour time, *MM* is the minutes, and *SS* is the seconds.

c. Input Parameters

The processing program parameters listed in Table 1 are described below. If the *read_expand_ramp_f22_v3* program is run interactively, the program name is entered at the command prompt, and the user is prompted for the five inputs as follows:

```
Enter the number of records to read from the original file:
```

This is the number of time steps of the original *fort.22* file to use (*norig*). The user should know how many time steps are in the input file. Typically, the entire file would be read. If the number of records of the original file is unknown, a large number (e.g., 999) should be entered to read the entire file. If this is done, the total number of records to be written, as recorded in the log file and written to standard output, probably will be incorrect. (See the note below for further information on this topic.) As that number is merely for informational purposes, an erroneous value will not affect program operation or output quality.

```
How many background ("zero") records needed?
```

This is the number of zero-padded time steps that are prepended to the output file (*nback*). A reasonable number of zero-padded time steps, say, 15 days' worth, may be

desirable. This number depends on the length of the ADCIRC run and how much spin-up is desired. A value of zero can be specified if no zero-padding is desired.

```
How many copies of the 1st record needed?
```

This is the number of copies of the first original data record (i.e., time step) (*ncopy*). It is recommended that as many copies be made as are needed for the length of the ramped data. This allows the data values to gradually increase from zero (or background) to the full scale of the first record over a period of several hours, up to as much as one day. If no ramping is desired, *ncopy* can be greater than zero to extend the data that many time steps, or it can be zero to not extend the data (e.g., to simply zero-pad the beginning of the data set). If ramping is desired and *ncopy* is set to zero, the ramping function will be applied to either the zero-padded values (if any) and/or the original data, depending on the time step at which ramping starts and the interval over which ramping is applied.

```
How many ramped records needed?
```

This is the number of time steps over which ramping will be performed (*nramp*). As with the previous input, this number should cover up to 1 day, or possibly more, as model run particulars require. A value of zero can be specified if no ramping is desired.

```
Ramped records will start at which record?
```

This is the time step at which the ramping will commence (*nstart*). If ramping is not desired (i.e., *nramp*=0), this prompt is not issued. The value of *nstart*, along with the others, should be determined in advance so that the beginning of the ramped data starts immediately after the zero-padded data (if any), and the end of the ramped data coincides with the end of the model spin-up period. Contact the primary author (Blain) for further details.

To obtain an estimate of the number of records in a *fort.22* file, one can use the Unix *tail* and *grep* commands. First, enter:

```
# tail fort.22
```

to see the number of elements for the last record (the 1st column on the last line of output). Then, to obtain the number of records, enter:

```
# grep -c ' NNN ' fort.22
```

where `NNN` is the number of elements (e.g., 121) displayed in the last line of the output of the *tail* command. The spaces inside the quotation marks are needed to obtain a correct count.

7. Reference

Garratt, J. R., 1977. Review of drag coefficients over oceans and continents, *Monthly Weather Review*, 105, 915-929.

APPENDIX I. ADCIRC fort.22 File Information

Fort.22 **File Format (source: *adcirc.org*):**

The first two records are written using an implicit do-loop (examples from the Fortran 90 code found in the data reader *navo_gmt.pm*):

```
write(6,*) (i,wsx(i,k),wsy(i,k),pr(i,k),i=1,np)
```

Example output format (first or second record):

```
           1  1.1568529E-05  3.5584922E-05   100431.9              2
   1.1568529E-05  3.5584922E-05   100431.9              3  1.1568529E-
05
   3.5584922E-05   100431.9              4  1.1568529E-05  3.5584922E-
05
    100431.9              5  1.1568529E-05  3.5584922E-05   100431.9
           6  1.1568529E-05  3.5584922E-05   100431.9              7
   1.1568529E-05  3.5584922E-05   100431.9              8  1.1568529E-
05
   3.5584922E-05   100431.9              9  1.1568529E-05  3.5584922E-
05
    100431.9             10  1.1568529E-05  3.5584922E-05   100431.9
```

The remaining records are written using an explicit do-loop:

```
do i=1,np
  write(6,*) i,wsx(i,k),wsy(i,k),pr(i,k)
enddo
```

Example output format (third record and higher):

```
           1 -1.4711546E-05  3.2182335E-05   102322.4
           2 -1.4711546E-05  3.2182335E-05   102322.4
           3 -1.4711546E-05  3.2182335E-05   102322.4
           4 -1.4711546E-05  3.2182335E-05   102322.4
           5 -1.4711546E-05  3.2182335E-05   102322.4
           6 -1.4711546E-05  3.2182335E-05   102322.4
           7 -1.4711546E-05  3.2182335E-05   102322.4
           8 -1.4711546E-05  3.2182335E-05   102322.4
           9 -1.4711546E-05  3.2182335E-05   102322.4
          10 -1.4711546E-05  3.2182335E-05   102322.4
```

where

np	is the number of nodes in the mesh
wsx, wsy	are the x,y wind stress components at node i, time record k
pr	is the surface pressure at node i, time record k

Description of Wind and Pressure Data Read in from UNIT 22

The input variables are described in Table Ia and are listed below by the line sequence

within the file:

```
IF(NWS.EQ.1)   NHG,WSX2(NHG),WSY2(NHG),  PR2(NHG),  NHG=1,NP
IF(NWS.EQ.2)   NHG,WSX2(NHG),WSY2(NHG),  PR2(NHG),  NHG=1,NP
IF(NWS.EQ.3)
  IWTIME
  WSPEED(I,J)
  WDIR(I,J)
IF(NWS.EQ.4)
  NHG,  WSX2(NHG),WSY2(NHG),  PR2(NHG)
```

Table Ia. Description of the Variable Names Associated with the ADCIRC fort.22 File

Variable	*Type*	*Description*
NWS	Integer	Wind stress and surface pressure option parameter in *fort.15* (See below table for further notes).
NHG	Integer	Node number
PR2(NHG)	Real	If NWS = 1 or 2, Applied atmospheric pressure at the free surface (N/m^2 = PA) divided by the reference density of water divided by gravity. If NWS = 4, Applied atmospheric pressure at the free surface (millibars).
WSX2(NHG)	Real	If NWS = 1, or 2, Applied horizontal free surface stress in the X-direction divided by the reference density of water at the node (should be units $(length/time)^2$). If NWS = 4, Applied horizontal wind velocity (knots) blowing toward the + X-direction.
WSY2(NHG)	Real	If NWS = 1or 2, Applied horizontal free surface stress in the Y-direction divided by the reference density of water at the node (should be units $(length/time)^2$). If NWS = 4, Applied horizontal wind velocity (knots) blowing toward the + Y-direction.
IWTIME	Integer	If NWS = 3, Time of the wind field in the following integer format: YEAR*1000000 + MONTH*10000 + DAY*100 + HR.
WSPEED(I,J)	Real	If NWS = 3, Wind speed in meter/sec.
WDIR(I,J)	Real	If NWS = 3, Direction wind blows from in degrees CW from north.

Table Ib describes some notes on using the ADCIRC wind forcing specification parameter, NWS. Details on NWS are provided following the table.

Table Ib. Notes on the Use of the Wind Forcing Specification Parameter, NWS

Variable	*Type*	*Description*
NWS	Integer	Wind stress and surface pressure option parameter.
		= 0 No wind stress or surface pressure is applied.
		> 0 Spatially varying wind stress and surface pressure are applied.
		= 1 Wind stress and pressure are read in at all grid nodes every time step from UNIT 22.

		= 2 Wind stress and pressure are read from the UNIT 22 file at all ADCIRC grid nodes at a wind time interval that does not equal the model time step. Interpolation in time is used to synchronize the wind and pressure information with the model time step. The wind time interval must be specified later in the UNIT 15 file.
		= 3 Wind velocity is read in from an outdated U.S. Navy Fleet Numeric format wind file on UNIT 22. This data is interpolated in space onto the ADCIRC grid. Interpolation in time is used to synchronize the wind information with the model time step. Garret's formula is used to compute wind stress from velocity. Several parameters describing the U.S. Navy Fleet Numeric wind file must be specified later in the UNIT 15 file.
		= 4 Wind velocity and pressure are read in at selected ADCIRC grid nodes from a PBL/JAG format file on UNIT 22. Interpolation in time is used to synchronize the wind information with the model time step. Garret's formula is used to compute wind stress from velocity. The wind time interval must be specified later in the UNIT 15 file.
		= 10 10 m high wind velocity and surface pressure are read in from NWS AVN model MET files. These files are in binary and have been created from a larger GRIB format file using UNPKGRB1. Each file is assumed to contain data on a Gaussian longitude/latitude grid at a single time. Data consecutive files are assumed to be separated by 6 hours in time.
		=11 10 m high wind velocity and surface pressure are read in from stripped down NWS ETA-29 MET files. These files are in binary.

NWS = 1

- The first data set is provided at TIME=STATIM+DT. Subsequent data sets are provided at every time step.
- Data must be provided for the entire model run, otherwise the run will crash.

NWS = 2

- The first data set is provided at TIME=STATIM. Subsequent data sets are provided every wind time interval.
- Data must be provided for the entire model run, otherwise the run will crash.
- The wind time interval must be set in the UNIT 15 file.

NWS = 3

- The first data set must be at or before the date and time listed in the UNIT 15 file as the beginning time of the simulation.
- Data sets are provided every WTIMINC, where this parameter is the wind time interval and is specified in the UNIT 15 file.
- Data must be provided for the entire model run, otherwise the run will crash.
- Values for NWLAT, NWLON, WTIMINC, the beginning time of the and several other parameters must be set in the unit 15 file.
- The following transformations are preformed to put this information into usable form for the model calculations.

```
WIND_STRESS = DRAG_COEFF*0.001293*WIND_VEL*WIND_SPEED
DRAG_COEFF = 0.001*(0.75+0.067*WIND_SPEED)
IF(DRAG_COEFF.GT.0.003) DRAG_COEFF=0.003
```

NWS = 4

- This line must have the format I8,3E13.5.
- This line is repeated for as many nodes as desired.

- A line containing the # symbol in column 2 indicates new wind and pressure fields (i.e., values at the next time increment) begin on the following line.
- Each node that is not contained in the UNIT 22 file is assumed to have zero wind velocity and pressure = 1013.0.
- The following transformations are preformed to put this information into usable form for the model calculations.

  WIND_VEL (M/S) = WSX2*1.04*0.5144, WSY2*1.04*0.5144
  WIND_STRESS = DRAG_COEFF*0.001293*WIND_VEL*WIND_SPEED
  DRAG_COEFF = 0.001*(0.75+0.067*WIND_SPEED)
  IF(DRAG_COEFF.GT.0.003) DRAG_COEFF=0.003
  PR2*100/GRAVITY/1000.

- The first data set is provided at TIME=STATIM. Subsequent data sets are provided every wind time interval.
- Data must be provided for the entire model run, otherwise the run will crash.
- The wind time interval must be set in the UNIT 15 file.

APPENDIX II. Input Parameter File Example

All necessary input parameters are listed in the input parameter file, *makef22.pl.in*. This ASCII text file contains values for all parameters, although some may not be used during the current execution of the program. The parameters are described in Table 1, §2b. An example of the file is presented here. Each parameter is on a single line, including the "`readeropts`" parameter. The parameter file name, the column after the "#" symbol in the third and subsequent lines below, must be included, and must be surrounded by at least one space.

```
#  makef22.pl.in -- Input parameter file for makef22.pl.
#  Custom versions of this file must use the same format.
2004011200          # start - Starting date/time (YYYYMMDDHH)
2004011400          # end - Ending date/time (YYYYMMDDHH)
6h                  # inc - Time increment
fort.14             # f14 - fort.14 file name
~makef22pl/readers  # readerdir (*.pm direct.) [OPTIONAL]
navo_gmt            # reader - Source code (Perl module, e.g., navo_gmt.pm)
gmt2f22_navo_gmt    # readerbin - Binary executable rdr. Program [OPTIONAL]
-O PW -domain CENT_AM -datadir /scr/ooc  # readeropts - rdr. options
i686                # arch - Architecture [OPTIONAL]
1                   # ramp - Flag to perform ramping
7                   # norig - Number of original time steps to read
11                  # nback - Number of background records to prepend
6                   # ncopy - Number of 1st original records to copy
6                   # nramp - Number of time steps to apply ramping
11                  # nstart - Time step at which ramping starts
```

In this example, the value of "ramp" is 1, so that extension and/or ramping is to be performed on an existing *fort.22* file. See §6 for details on this processing option.

APPENDIX III. Details of the Date Iterator

The module responsible for stepping through a date sequence was written specifically for applications such as *makef22.pl* that are meant to read/manipulate data identified by a date-time stamp. *Iterator.pm* is not dependent on any non-core Perl modules which makes it portable to any system that has Perl version 5.6 (or greater) installed.

As an aside, any application in Perl that requires iteration over a range of dates can use this module. Its application interface is designed to be straightforward and easy to use.

Here is an example Perl application that uses the *Iterator.pm* module:

```
#!/usr/bin/perl –w     # shebang line identifies interpretor for script
use strict;            # Perl "strict" mode enforces strict code rules
use lib qw(./);        # look in './' directory for 'use'ed modules
use Iterator qw(:All); # use Iterator(.pm) module

my $iter = Iterator->new();    # create new Iterator object
$iter->set_start('2003100800); # start date, YYYYMMDDHH required format
$iter->set_end('2d');          # end date, relative or YYYYMMDDHH date
$iter->set_increment('12h');   # date increment, use y,m,d,h or combin.
$iter->set_format('%Y%m%d%k'); # format of date string – consistent with
Unix `date` utility

# Loop over date range

while ($iterator->next()) {
  # do something
  my $current_date = iter->get_current();
  print "Iteration for date $current_date\n";
}

### THAT'S IT!
```

Resulting Output:

```
Iteration for date 2003100800
Iteration for date 2003100812
Iteration for date 2003100900
Iteration for date 2003100912
```

Table II describes the functions used by *Iterator.pm* in the provided example.

Table II. Description of Functions Used by Iterator.pm

Method	*Req*	*Description*	*Valid Values*
new	Y	Creates new instance of iterator object	N/A
set_start	Y	Start date	YYYYMMDDHH – this format is required

Method	*Req*	*Description*	*Valid Values*
set_end	Y	End date; an actual date can be specified or a relative period of time after the start date.	YYYYMMDDHH, or any combination and order of: #h – hours past start, #d – days after start, #m – months after start, #y – years after start For example, “2m3d4h” will end 2 months, 3 days, and 4 hours after the start date.
set_increment	Y	Time increment; the time interval of the data records	Any combination and order of: #h – hours, #d – days, #m – months, #y – years For example, “2m3d4h” will increment 2 months 3 days and 4 hours for each iteration.
set_format	[N]	OPTIONAL: Set format using macros (see right)	%B January...December %b Jan...Dec (same as %h) %d padded day of month %e day of month (no padding) %h Jan...Dec (same as %b) %k padded hour of day; 00-23 %l hour of day (no padding) %m padded month; 01-12 %Y four digit year %y two digit year; 00-99
next	[N]	OPTIONAL: Increases current date by the specified increment of time	N/A
get_current	[N]	OPTIONAL: Returns the current date in the specified format.	N/A

APPENDIX IV. Plug-in Reader Application Programming Interface

The functions shown in Table IV are required in order to create a basic data reader for *makef22.pl*. Other functions can be added to the reader as desired, but these four functions are called explicitly by *makef22.pl*. Otherwise, a non-existent function call will result in an error that will cause the program to terminate.

Table IV. Required Plug-in Reader Functions

Function	***Parameters***	***Description***
`new`	String	Initializes the reader and passes reader options defined by the string in *makef22.pl's* "`readeropts`" parameter to the reader for parsing.
`print_info`	None	Prints information about the reader to the screen after initialization .
`get_record`	Date string of format "YYYYMMDDHH"	Returns the *fort.22* record for this date/hour .
`finalize`	None	Performs any remaining tasks after iteration through the entire date range has been completed.

APPENDIX V. Supported Domains in the Data Readers

Currently, the NAVO GMT data reader, *navo_gmt.pm*, supports ten geographical domains for which the COAMPS model is executed operationally. These are listed in alphabetical order of the alias name in Table Va. Details about the domains can be found in §4b and Table 3.

Table Va. List of the Geographical Domains Supported by navo_gmt.pm

Domain Alias	*Generating Model*	*Supported Fields*
ARAB_SEA	COAMPS	wind, wind stress, pressure
CENT_AM	COAMPS	wind, wind stress, pressure
E_PAC	COAMPS	wind, wind stress, pressure
EUROPE	COAMPS	wind, wind stress, pressure
N_IND	COAMPS	wind, wind stress, pressure
SOUTHWEST_ASIA2	COAMPS	wind, wind stress, pressure
SOUTHWEST_ASIA3	COAMPS	wind, wind stress, pressure
W_ATL	COAMPS	wind, wind stress, pressure
W_PAC	COAMPS	wind, wind stress, pressure
W_PAC2	COAMPS	wind, wind stress, pressure

The NRL COAMPS data reader, *nrl_coamps.pm*, also supports ten geographical domains for which COAMPS model output are archived. These are listed in alphabetical order in Table Vb. Details about the domain structure in the reader are described in §5b and Table 6.

Table Vb. List of the Geographical Domains Supported by nrl_coamps.pm

Domain Alias	*Generating Model*	*Supported Fields*
CEN_AMERICA	COAMPS	wind, wind stress, pressure
E_PAC	COAMPS	wind, wind stress, pressure
EUROPE2	COAMPS	wind, wind stress, pressure
N_IND	COAMPS	wind, wind stress, pressure
N_IND2	COAMPS	wind, wind stress, pressure

Domain Alias	*Generating Model*	*Supported Fields*
SW_ASIA2	COAMPS	wind, wind stress, pressure
SW_ASIA2_N3	COAMPS	wind, wind stress, pressure
W_ATL	COAMPS	wind, wind stress, pressure
W_PAC	COAMPS	wind, wind stress, pressure
W_PAC2	COAMPS	wind, wind stress, pressure

The NRL NOGAPS reader, *nrl_nogaps.pm*, supports three geographical domains for which NOGAPS model output are archived. These are listed in alphabetical order in Table Vc. Details about the domain structure in the reader also are described in §5b and Table 6.

Table Vc. List of the Geographical Domains Supported by nrl_nogaps.pm

Domain Alias	*Generating Model*	*Supported Fields*
NOGAPS0_5g	NOGAPS	wind, wind stress, pressure
NOGAPS1_0g	NOGAPS	wind, wind stress, pressure
NOGAPS1_0 (older files)	NOGAPS	wind, wind stress, pressure

Details about the NRL COAMPS and NOGAPS atmospheric model data can be found at the web site http://www7320.internal.nrlssc.navy.mil/CANS/index.php which lists the model domains, the parameters generated by the models, model time steps, and other pertinent information. To obtain a list of the currently supported domains, change to the readers directory where the .pm files reside and issue the following command from the command line:

```
grep ABBREV *.pm
```

which will list all of the domains available in the readers. The output will look something like the following:

```
navo_gmt.pm:          ABBREV => "CENT_AM",
navo_gmt.pm:          ABBREV => "E_PAC",
```

and may contain additional domains from this and any other reader files.